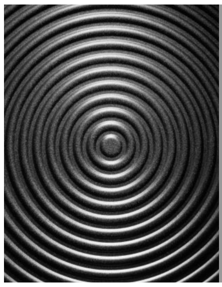


CHAPTER 1

Metadata, Resources, and the Resource Description Framework

This book is about a technology that lets you tailor and filter content; create ad hoc portals; build profiles of users, information services, and objects; and generally give the user a richer, custom-fit experience. It is about how you can build metadata systems, especially systems that use profiles to create information services. Profiles, of course, are nothing but structured metadata descriptions, so they are not really remarkable in themselves. The remarkable thing happens when you start applying profiles to create services.

The Web is less than ten years old, and chat rooms are rife with complaints about the World Wide Wait. While there are some companies who deliver excellent service, most do a really bad job of it. The reason is not just that they do not understand what adds value, it is also a consequence of the enormous amount of data on the Web.

You may well wonder what this has to do with wireless. But wireless access to Web information has proven to be not just the trigger for new types of devices and presentation formats, but for new types of information services as well. There is also a theoretical foundation: The more restricted the properties of a device, the less complicated grammars it can handle. And RDF is a very simple grammar.

The way we use wireless devices is profoundly different from the way we use desktop computers, even though the information is the same. Building transactional services on top of the information requires that you know more about both the information and the user than the services geared toward passive browsing. This is something we already see on the Web, but the real catalyst has been iMode in Japan and Wireless Application Protocol (WAP) in Europe, which have set off a landslide of creativity among developers.

Metadata enables search engines to get better results with fewer hits, making searches more precise and tailored to the users' needs. Metadata enables personalization—the current silver bullet of marketing prophets. Metadata lets you filter out information you do not want (or that you do not want to reach someone). It is really simple to add to a Web site. What is missing is software and services that use it, but those are coming too.

However, this way of using metadata is scary to old-economy Chief Information Officers (CIOs) and Web designers. You are giving control to the users. You are giving information away, and it might decrease the number of hits on your pages.

If your company still measures success on the Web in number of hits, fire the Webmaster and the CIO. What matters is not how many people pass your store, but how many people enter and buy. Fewer users may get the information, but they are the right users, and the conversion rate among them will be higher. The irritation of users who get your page when they were searching for something completely different will also decrease, increasing your goodwill by default.

Metadata is about relationships. It is about descriptions of resources, which in this context are things which provide services: servers, database engines, Webcams, anything that is providing information. When you write a table in HyperText Markup Language (HTML), you have a set of relationships in mind that describes the rows and columns. If this could be formalized in a machine-readable language, the system could use those relationships, too. HTML version 0.9 was very poor in most respects, compared to HTML 4.0. Yet it contained the core of what was necessary to start off the industry. In many respects, Resource Description Framework (RDF) is the same today.

The Web today runs on HTML, but it has become an old technology that can only take you so far. You need new technology to enable new services. Indeed, the traditional Web is proving to be a legacy that is hard to overcome. The way information services are used and deployed in the new environment requires a different kind of service than that which is available on the traditional Web: a type of service that is more convenient, easier to use, and faster than the old Web. And, of course, usable and useful on the old Web as well.

RDF was developed at the intersection between the knowledge management world and the library metadata world. It is a graph system layered on top of Extensible Markup Language (XML), and thus has two roots: Directed graphs, which are probably more familiar to database programmers; and XML, which is certainly familiar to the large contingent of programmers who have learned it in the last few years.

The XML aspect is important for several reasons. XML is as close to a global, universal data format as we come today (because it uses Unicode, it is more universal than ASCII, which is restricted to the Latin alphabet). How to handle it in databases, how to transport it over the network, and how to build applications that use it are all well understood. And it has a large installed base, with plenty of applications.

RDF builds on XML to create descriptions, and descriptions are metadata: data about data. It can be very hard to understand, and there are basically two ways of explaining it: As object properties, or as profiles. Which is more useful to you depends on your background.

First, let us start with the concept of descriptions. A description of a document is a document in its own right. Documents are nothing but a sequence of fragments, elements of information, and the order and structure of the fragments constitute a metadescription of the document. Of course, there can be other descriptions as well, such as what the document is about, how it should be presented in different formats, and anything that pertains to the document but is not the document itself.

In RDF, you always identify the object you are describing by a unique address, the Universal Resource Identifier. The descriptions can be object identifiers as well (URIs). This means that in object-oriented terms, you can describe which classes an object belongs to—and then compare the listing of classes for this particular object to other objects—and so find out what they have in common.

The descriptions are also data. Data about data is data, too. And metadata is nothing but data about data. It can be embedded in the document, or exist separately from it, as a document or as headers in a protocol, for instance. But there is nothing that limits the use of metadata to documents. It is possible to describe any object using metadata. And anything can be an object, from the collection of all information in the universe down to the letters on this page.

When the metadata about the object is structured to provide a description, and the structure is common for all instances of the same type of object, it is a profile. The profile can have different values for different instances, but the structure is always the same. So, all the books in the library can be described using the same library cards, but each library card will have different content, even though they all include book title, author, and so forth.

What falls under one classification to one man, however, is something else to another. No object falls unambiguously into a single classification. Not even in physics can we find unambiguous ways to describe the objects we talk about. What is a quark? Is an electron really a wave or a particle? And it just keeps going uphill from there.

Language, from the point of view of the language philosopher, is based on a social contract between the sender and receiver. If I am speaking Japanese and you know only English, you will not receive the message I am sending. If I am speaking Japanese grammar with English vocabulary, your understanding will also be seriously impeded. There has to be agreement about the syntax (the combination of the words), the vocabulary (the mapping between symbol and concept), and the semantics (the meaning of the vocabulary). RDF is a set of rules for creating semantics, and RDF Schema is a way of creating vocabularies.

In practice, however, you rarely need to get that philosophical. You do have to recognize that classifications are necessarily arbitrary and that the names of concepts are inherently meaningless. Any knowledge representation scheme will have to take this arbitrariness into account.

Representing knowledge in a computerized format has been high fashion in the computer industry for a long time. Knowledge representation systems are an important aspect of metadata, but they have a big inherent weakness. While they are mostly geared toward the scientific domain, they assume they are general, both in vocabulary and structure. Of course, they are not, and while they have come up with a lot of valid results in the field of intelligent agents, the work still suffers from the assumptions that there are universal ways of representing knowledge within one, centrally determined framework. This is most evident in the classification schemes used in the library world. Classifying, as they do, information objects that represent anything, they do have presumptions to be general. But they are really specific, not in the least culture-specific. A library in Sweden is not classified in the same way as a library in the United States.

RDF addresses this by providing for a decentralized scheme. Using the same structure, it allows anyone to create their own vocabulary. This enables you to take the good things from the artificial intelligence community (intelligent agents and reasoning systems) and apply the good things from the Web (the transport system, the data representation, the decentralization and independence from a particular system). This intersection is the topic for this book.

What Is RDF?

RDF is a format to make assertions—statements that are intended to point something out. It has two roots: metadata and knowledge representation. The concepts do overlap, but the technologies which have been developed in the two fields, and the understanding of them, are very different.

Library communities have struggled to come up with a universal format for describing books that can be used in electronic catalogs. The result is the Dublin Core format (named after Dublin, Ohio, and not Dublin, Ireland). It is basically a set of attribute names and rules for which values they can take. The attributes are limited to those that make sense in the world of books, and not for a more general set of knowledge representation, or even for magazines.

The knowledge representation community, on the other hand, was the hot thing of the late 1980's, expected to become the next huge thing in the software industry. It fizzled for different reasons, but they came at the problem from a different direction. An expert system is a set of rules, into which you can fit knowledge as input, and get different knowledge sets as output (for instance, if the input is the symptoms of a patient, you can get a recommendation for treatment as output). But this means that you want to find a way of representing any and all knowledge that fits into the system.

This brings a different set of problems. If you just think about metadata within one single domain, it makes sense to have a table with the attribute names in one column, and the attribute values in the other. But if you want to compare the attributes to something else, how do you know what they represent? What is the unit of measurement if thickness is 1, for instance? Inches or centimeters? Or millimeters? Or meters? Or miles? This seemingly simple problem made the Mars Surveyor crash, and it can easily make someone else misunderstand what you want to present, especially if you want to present it to a computer program. A computer is stupid, and if you do not tell it exactly what you want to say, you cannot expect the receiving software to take any relevant actions.

RDF, the Resource Description Framework, was developed by the World Wide Web Consortium (W3C) to create a format for making assertions that leverage the XML format to represent and transport information. XML, however, is not a markup language; it is a set of rules for creating markup languages. I will talk more about this in Chapter 2, but suffice to say that using XML brings us as near to a universal data format as is possible nowadays.

However, XML gives only the rules for how the byte strings should be cobbled together to form a coherent whole, which can be used by a widely spread set of computer programs. How an XML data set should be interpreted is determined by the Document Type Description (DTD), which essentially creates a schema for an application of XML. But XML does not say anything about the information itself, only the way it is structured.

The level above XML determines how the information is interpreted, and this is where RDF exists. It builds to a very large extent on experiences from the knowledge representation community, creating a way of comparing different knowledge representations by making the user define them in a specific way and enabling the comparison to use a format which is defined in a branch of mathematics called graph theory. RDF is a way to express relations between objects, something XML does not allow you to do.

Describing Data: The Concept of Graphs

A relation between two objects can be described in many ways; one way is to draw on a piece of paper two circles representing the objects, and a line between them representing the relation. This is actually the beginning of a well-established analysis method for applications, using Unified Modeling Language (UML) to describe the relations. However, once you have established a description of the relationships, it is possible to do much more with them.

A description of a Web site is like the morning promenade of a German philosopher (not because "neither can whistle"). These two far-flung concepts are connected in that the same theory can be used to relate the description of one Web site to another.

First, the description of the Web site. There are a seemingly (if not actually) infinite number of ways to describe a Web site. However, they all have one thing in common: Each describes a Web site. (In case it describes an object that is not a Web site, the logic is the same.) The descriptions can be broken down into logical elements, as follows: Instead of writing "this is my really fun site that talks about lots of great stuff that I found in other places on the Web and that I really liked," you can write "site has property=fun (according to: you); property=links (to: stuff from the Web, appreciated by: you)." You are describing an object (the site) that has properties (fun, links) that have values (that you think it was fun, that the stuff is from the Web). So far, anyone familiar with object-oriented analysis will not have discovered anything remarkable.

If you draw your description on paper, you can draw it as dots connected with lines: In mathematics, this is called a graph. Since the description can be thought of as a graph, the branch of mathematics that includes graph theory—topology—can be applied to it. I will take this reasoning a little further, but I will not try to give you a real introduction into graph theory, because that would go too far.

The simplest graph you can imagine is two dots connected with a line. If the dots represent something, they can be nodes, and the line connecting them an arrow. If the line is an arrow, it has a direction. The Web is a directed graph in which the nodes are documents, and the arrows are the links between them (because a link has a single direction, at least in HTML). As a matter of fact, the entire Web is a big directed graph, because the links are unidirectional (they go from one document to another). This is changing with the introduction of bi-directional links in HTML 4.0 and Xlink/Xpointer, but the Web as you know it is a one-way street. Links go only forward, not back.

If you add attributes to the arrow—called an arc in mathematician-speak—you are creating a labeled directed graph. This is the format that RDF uses to describe resources.

A set of documents which has a set of properties can be described as nodes in a network, with arrows (properties) pointing at the values of the properties. Since each property applies to only one object, this is a directed graph, a diagram that you can follow in one direction only. The arrows do not have to go in one direction (but there has to be one arrow per direction). For a simple graph, this is not complicated. Here is what the statement, “Napoleon was emperor,” looks like as a directed graph:

```
Napoleon → emperor
```

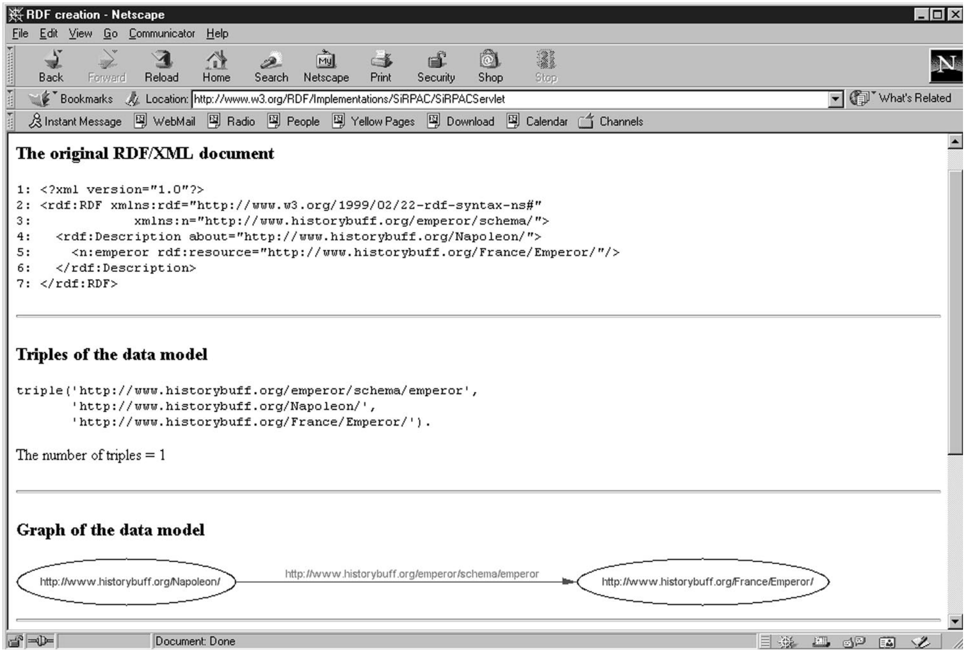
(As a matter of fact, I am cheating. What it says is “Napoleon has the property Emperor.” It says nothing about the temporal aspect, that he actually is no longer emperor). If you are familiar with object-oriented programming, you will see that this is the same as the Booch methodology analysis statement that the object Napoleon has the property Emperor. And indeed, RDF is a format to describe the properties of objects, which are identified by their Uniform Resource Identifiers (URIs). You do not, however, have to subscribe to any of the mysticism often surrounding object-oriented programming. The objects here are not programming objects, they are information objects, which is different. Information objects do not follow the same rules as programming objects. For instance, they cannot do anything (they do not have methods associated with them), and the inheritance of properties between classes is not at all as clear-cut.

RDF is primarily intended to describe information objects, but it is not really restricted to that. It can be used to describe anything that has a URI, which in principle can be any object. And because objects are related to our perception (“What constitutes an object?” is a question that we will not go into here, but it is one that philosophers struggle with), anything we can perceive or imagine can be an object. And provided it can be given an identifier, it can be described mathematically. Provided that identifier can be a URI, it can be described in RDF, as it can then be a resource in the sense of RDF.

The graph above can also be formulated as a triple of values: The resource being described, the property name, and the property value. RDF describes how to take the triples into XML, and how to do this in a way that maintains the integrity of the graph.

The W3C runs an excellent validation service based on the Simple RDF Parser and Compiler (SiRPAC). Figure 1.1 is an example of what the graph would look like, as well as the RDF associated with it.

To a mathematician, the paper drawing and the mathematical formulation of the graph are the same thing, just formatted a little differently. This means that to a mathematician, an XML encoding of the graph is just another form of expressing it. You can interchange the different description formats; they are


RDF creation - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://www.w3.org/RDF/Implementations/SiRPAC/SiRPACServlet
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Document: Done

The original RDF/XML document

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:n="http://www.historybuff.org/emperor/schema/">
  <rdf:Description about="http://www.historybuff.org/Napoleon/">
    <n:emperor rdf:resource="http://www.historybuff.org/France/Emperor/"/>
  </rdf:Description>
</rdf:RDF>
```

Triples of the data model

```
triple('http://www.historybuff.org/emperor/schema/emperor',
       'http://www.historybuff.org/Napoleon/',
       'http://www.historybuff.org/France/Emperor/').
```

The number of triples = 1

Graph of the data model

$$\begin{tikzcd}
\boxed{\text{http://www.historybuff.org/Napoleon/}} \arrow[r, "\text{http://www.historybuff.org/emperor/schema/emperor}"] & \boxed{\text{http://www.historybuff.org/France/Emperor/}}
\end{tikzcd}$$

Figure 1.1 The statement, “Napoleon is emperor,” in RDF.

equivalent from a mathematical standpoint. And because programming is a way of creating mathematical models of actions, this also means that to a program, they are equivalent.

To a mathematician, graphs are very rarely unique. They can be transformed into other graphs, and operations can be conducted on them. Algebraic operations are used to model computer software. This means that there is a well-established way of going from the graph format of RDF to a way of handling it in the computer.

The more you want to say, however, the more complex it becomes. A graph does not have to be two-dimensional, either; it can have *n* dimensions and be arbitrarily complex. Of course, this slows down processing and decreases the readability. But it is possible to express quite complex relationships as graphs. Here is a graph of the statement that Napoleon first was a lieutenant, then a general, then a consul, and finally emperor:

$$\begin{tikzcd}
\text{Napoleon} \arrow[r] & \text{lieutenant} \arrow[r] & \text{general} \arrow[r] & \text{consul} \arrow[r] & \text{emperor}
\end{tikzcd}$$

Of course, the graph could be drawn differently. This graph actually does not refer back to Napoleon, but refers from one concept to the other.

There is an entire mathematical theory associated with graphs, how they should be handled, what transformations are allowed and possible, and so on. The operations you can do on RDF descriptions are described in graph theory.

Since the XML description is just another way of expressing a graph, it is possible to do exactly the same operations on the XML description as on the graph drawn on paper. It is somewhat complicated to go back to the graph from the XML representation, however, because the same XML representation can be drawn as several different graphs (all graphs which have the same format are the same to a mathematician).

Because you are working with triples of information that contain relationships to each other, you can also filter out information with a much higher degree of precision than alternate technologies. If you have an attribute-value-table, you can filter out information based on the value of the attributes. If another site does not share the same attributes and values, you cannot use information from that site in your filtering process. However, if both your site and the other site express the information about yourself in a format that relates the statements to you, you will be able to filter out information to create a much more precise selection of information for the user. This can, for instance, be applied to a site which creates a home page from two different sites (what might be termed a personal portal) based on the user's preferences. It can be used to describe the copyright restrictions you have set on the use of the information, and it can be used to describe the relationship of your site with the site providing the personal portal.

As you can understand, whether you call the arcs *Evidence A* or *a horsetail* does not matter to the underlying graph. However, the properties of the elements can change the graph. You need to declare what elements you are going to use, and which properties they are going to have. These properties will change the way you can make statements. They will change how the terms interrelate, both in the current graph and with elements in other graphs. You need to declare the elements and their properties, and the structure in which they can occur, the schema of the graph.

Schema in the present context refers to a data structure. It has gained that use in the database industry, where it describes the structure of a database table: the columns and their headings, and what restrictions there may be on the data in the columns.

Both the properties and the RDF elements are objects in their own right, and can be the subjects of statements in themselves. This means that there can be graphs chained to the graph you look at orthogonally; or in any number of dimensions, because to a mathematician, the paper drawing is not necessarily the clearest description.

RDF Vocabularies

A schema does not define only the data structure for your graph. It also defines the terms you can use in it, the vocabulary of the description. For instance, the schema provides property names and property values, as well as constraints on them that enable their use in calculation. To a computer, a vocabulary is just a set of objects with specialized properties: one word can only be a figure, another can only be five characters long, and so on. A computer cannot have a relation to the vocabulary as such. It is silly that I have to point this out, but the anthropomorphization that follows on the use of *intelligent* for agents means that users tend to ascribe to them properties that they do not have. Large parts of this book will be devoted to demystifying the relationship between agents and data.

The artificial intelligence field is important in many aspects, not in the least because it serves to identify the problem space. There are four problems that, in translation from concept to practice, have dogged intelligent agents:

- Data encoding
- Data transport
- Resource discovery
- Drawing conclusions based on data

The artificial intelligence work has largely concentrated on the last one, and no practical systems exist that solve the first three problems in a distributed fashion. Using the technologies developed on the Web, however, all four of the problems become simple. Applying technologies from the World Wide Web to artificial intelligence systems solves problems that have dogged researchers for the last twenty-five years, at least.

There have been a number of vocabularies that attempted to describe specialized domains before RDF was invented. Most of them are specialized vocabularies defined for an area of artificial intelligence. There are a few examples of other types of vocabularies where an attempt has been made at formalizing a terminology. (There are probably as many terminologies as there are disciplines, and most specialized domains of knowledge have their own terminologies: Engineers talk about nuts and bolts in a particular way, whereas a theater director will have a precise vocabulary to describe the action on the stage.) The examples include the rules for games, which are formalized very precisely to determine which types of clubs are allowed in golf, or how you can touch the ball in soccer. These rule books contain very precise definitions of the different aspects of a game. But they are not concerned with information objects; if we were talking about a sports service, it might make sense to define the vocabulary in terms of the rule book for the game, for instance.

RDF gives you a format for describing information objects, but it does not say anything about what terms you should use to make the statements that describe the objects, and what those terms mean. It does, however give you a format for the terms and their formalized descriptions. It gives you the graph structure, but not the names and properties of the nodes.

Knowledge is always inherently based in the experiences of the user, and if it is expressed in a formal structure, it is important to describe it in such a way that one instance of a term does not have a completely different meaning than another. Here, we come back to the ambiguity of descriptions. For instance, the term *suit* means *a set of clothing* in one case, and *a set of playing cards bearing the same symbol* in another; it is very hard to know which suit I am talking about if I do not declare which domain it is in. In conversation, I usually do so implicitly. But if the word occurs without any supporting information, there has to be a declaration about what it is supposed to mean. Once I have explained to you that a suit is a piece of clothing, you will go happily to court to receive your suit. You, however, know that in a courtroom, a suit is not a piece of clothing, but a lawsuit. A computer does not. It will assume that they are the same thing, if you do not tell it otherwise. Remember, computers are really stupid, and one of the biggest mistakes of the artificial intelligence movement was to overestimate the intelligence potential of computers and to underestimate the enormous amount of learning required to think like a person.

Because computers are stupid, you have to be very careful to define terms in a totally unambiguous way. In RDF, the declaration of the terms you will use in the assertions (the statements about the objects) is done in a separate document called a RDF Schema, and in a specific language (the RDF Schema language). The schema also has to be accessible at the URI you use to declare the namespace for the elements you are using.

One of the interesting things about RDF is that a central repository is not needed, because every time an unknown term is encountered, the RDF processor has to download the schema in order to make sense of the term. However, if there is an agreement among several actors to use a specific set of terms for a given domain of knowledge, the RDF processor will not need to look up a new schema every time, decreasing traffic in the network and speeding up processing. This is the case in the User Agent Profile (UAProf) vocabulary for CC/PP, for instance (a vocabulary describing client devices in the Composite Capability/Preferences Profile format).

Scenarios for Metadata

Metadata-based profiles can be used to change the services a Web site provides, enable new types of services, and allow developers to program presentation and tailoring of data. Metadata in libraries (the most common example) is used to find books; for example, you can search for the books in the card catalog by author or by classification (according to the Dewey decimal system, or some other system). But metadata does not have to be limited to one type of description. You can also find out which books are oversize and are placed on a separate shelf. Knowing which metadata will become the most useful is impossible to say in advance. To the librarian, the book classification is the most important. To the historian of the printing industry, the size of the book, its age, and the number of pages will be the points of interest. It is also hard to say what is data and what is metadata. The distinction may depend on how the data is used.

Now, regard the book as an object, and the information about it as an information object. A book has a number of properties, such as whether the book is available or not, which language it is in, where it is in the library, who has recently borrowed it, and so on. Those properties—the attributes of the book (what determines its *bookness*) and the values of those attributes—are all part of the information object.

Metadata applications are similar even when used in different ways. The underlying information structures are the same, irrespective of what is done with them when they are presented to the user. This means that there are potentially an infinite number of applications, which can cover any area in the universe. Of course, it is not possible to describe all of them in one book, but it is possible to describe the general principles behind them. And so, let us look at a few scenarios.

The Library Visit

The library community already uses metadata to a large extent; indeed, the library community has been driving the development of metadata technologies. The library card catalog is probably the best-known example of metadata in existence, and also the one with the best-established pedigree.

Lisa Simms, for instance, is studying at the University of Hawaii to be a teacher. Today, she needs to find a book by the French pedagogue Cèlestin Freinet, famous in Europe for his development of a participatory school system. She switches on her computer and looks up Freinet in the library search engine.

It turns out that there are no books by him in the library, so she turns to the university search engine, and at the same time types in her query into her regular search engine.

The regular search engine found a number of hits on phrenology, the "science" of divining people's characteristics by the bumps on their heads.

The university search engine, which classifies the sites by their content, does not list the phrenology sites; it does list a number of sites in French, which she cannot read. She curses herself that she did not think to turn on the language preference in her browser, so that it could match her language preference with the language of the Web site.

It turns out that the Graduate School of Education at the University of California, Berkeley, had a class on Freinet last year. Not only does the system tell her that one of his books is available in French from UC Berkeley and in English from UC Irvine, it also provides a link to a site set up by a student in the Berkeley class last year as part of a project to use computers in a Freinet school. The student's site was not even listed by Lisa's regular search engine, but she found all the information she needed because it was classified in such a way that the university search engine could find it.

Along with her search results, she gets to see an advertisement from the French bookstore chain FNAC to buy a Freinet book and have it shipped tomorrow, with the chance to win a trip to Paris for two. Tempted as she is, she would not be able to read it, so instead she borrows it from UC Irvine.

Dynamic Yellow Pages

Lara Mann is looking for a grooming service for her poodles. As a Los Angeles stylist, she realizes the importance of her dogs' image for her own image. She does not have the time to take care of them herself, however, so she needs to find a dog stylist close to her home. She takes a look in the yellow pages.

Under "Dog Care" there are a number of stylists, including three on the street where Lara lives. It does not say when they are open, however, nor whether they are any good. Two of them have very nice advertisements, but Lara had a horrendous experience with a hairdresser once, and she knows better than to trust advertisements. She is a member of the Poodle Owners Club of San Joaquin Valley, so she sends out an e-mail to her friends and asks whether they have any experience with the stylists near her.

While she is online, she also finds a couple of Web sites with poodle information, featuring consumer reviews of stylists. After reading a few of them (although they are all anonymous), she gets a really bad feeling about one of the stylists (who had the biggest advertisement). But when she gets the answers to her e-mails, the other ladies are overwhelmingly positive.

Poor Lara does not know what she should think. How is that stylist really? Whom should she trust? She ends up taking her poodles to one of the others, who colors them bright red instead of cool pink, and cuts Soviet flags into their coats as well as Stalin mustaches. She regrets not choosing the stylist she was doubtful about, who meanwhile has given her neighbors' poodle an incredibly tasteful treatment.

Of course, she can do all this today, but there is no standardized and simple way to link all the pieces together. How do you determine which e-mail talk to trust about which stylist? There is no way to way to link annotations to objects to get the opinions of others attached to the Web site.

Position-Dependent Information

Charles Rowland is lost in a foreign city. It's his first visit to the fascinating city of Yokohama, but he must have gotten turned around on his way from the gigantic Ferris wheel in the harbor to Izeyake Mall. In Japan, houses are numbered in the order that they are built, and there are no street names. Not much help. Looking at the signs on the street, he still cannot figure out where he is, because he cannot read a word of Japanese.

Luckily, he brought his PDA with a GSM receiver. Not only does it have a map with a big red dot in the middle showing where he is when he switches it on; he can also type in the name of the hotel, and the PDA shows him which direction it is. As he walks toward the hotel, the PDA continuously searches for sites on the Internet, compares their positional relevance with his position, and about lunchtime points him to a very nice little restaurant.

If there had not been any such descriptions, he would have been lost. And of course, he doesn't know that the map is generated from a description of the city and presented to him as a vector graphic, which makes it easy to zoom in, double-click on a famous place, and get a video clip.

Corporate Knowledge Representation

You have probably wondered who in your company is the great expert on a certain subject, or where on the intranet the configuration files for the video projector are. An intranet suffers from the same problems as the Internet: It is hard to find information, and when you do, you do not know what is relevant. Since business success is about getting an information advantage, finding information fast is critical to getting ahead of your competitors. And not just finding information, but finding the right information. Selecting the most relevant pieces in the enormous cloud of information that enters your company every day is critical to keeping it competitive.

On the other hand, in a company, this is easier to fix. There are no standard formats for data on the Internet, but you can easily enforce them inside your firewall. Or, as is likely to be more popular, you can enforce a specific metadata structure for certain types of information. This will enable your company to make sense of its own information faster, and relate it to external information better. Not only can it lead to enormous savings because you will not have to reinvent the wheel several times; it can also help you relate your information to that of your competitors, and discover how your company should develop to do a better job of servicing your customers.

Take Helen Farnsworth, who is a salesperson at historybuff.org, a company that publishes interactive games based on historical battles and other events. In reality, it is a division of major games maker Arcaditopia, and this is exactly why Helen is in trouble. She has a customer who is interested in their new game for a nationwide chain, but she cannot answer his questions, especially not about Elba. She can't answer mainly because the questions are about the historical accuracy of the game, and while she can tell the customer everything about the shelf space the game will take, and how much memory it will use, she does not know who beat Napoleon at Waterloo.

So she does a quick intranet search. The regular search engine gives her several hundred answers, among them Napoleon Jones in the accounting department. She searches for individuals with knowledge about Napoleon the historic person, and throws in Elba to make sure. Her meta-search engine helps her to find Herbert Wollmeyer III, the resident expert on the Napoleonic wars. And his "100 quick questions about Napoleon" tells her everything the customer asks about Elba (like, it's a part of Italy). And the customer buys 100,000 games.

The systems ribed in these quick scenarios get information that is customized in some sense. They get you an information set adapted not just to your personal profile (such as your credit rating), but also to the device—mobile phone, PDA, or PC—that you are using. Everything I describe above, you can get from the Web today. But you can get it only from isolated systems, and you can only use it for your PC, as a rule.

The implementation of these systems is very different from the traditional computer systems you are used to, however. They are generating information based on profiles, and creating optimized profiles based on the different profiles that are being received from the client, describing it and its situation. Using RDF, of course.

What Do You Need to Do?

As a site owner, there are a few things you need to do to create metadata for your site. As of today, there are few tools that let you insert the data automatically, especially if you are using XHTML. However, a summer intern and some clever programming will give you a metadata system that can be used in automating the insertion of metadata in your site. Meanwhile, there are a few things you can do:

Analyze your information. Analysis of information is a special discipline within knowledge management, with well-established methods and models. You do not need to go that far, however. It is enough if you do a quick and dirty subjective analysis (i.e., what do you think your site is?)

Determine what you want to say. Is your goal that people who download your pages should be aware of who created it, or is it that they should find you easier with a search engine? Or both? Is your target group other companies in your business, or do you want to reach anyone and everyone on the Web?

Classify your information. Remarkably few sites (even the do-gooders of the Web, the W3C) distinguish between officially published documents, e-mail lists that are archived, and temporary pages that are work items. Classifying your documents according to a pre-established vocabulary will let users find the documents on your site that are most relevant. *Determine which vocabulary suits you.* Once you have an understanding of you want to say, you can determine which formal vocabulary you should use to describe your site.

Create the statements. Having determined that, you can create the statements that describe your site. This can be done using any one of a number of different tools.

Insert the statements in your documents. Once you have created the statements, you can insert them into your documents (or templates, if you run a database-driven site). And then, users will be able to work with your information in a way that is much more satisfying to them than today.